```java
package AlejandroCastellanosExamenT11;

import java.util.Objects;
import java.util.concurrent.ThreadLocalRandom;

public abstract class Personaje implements Comparable<Personaje>{

    private final String nombre;
    protected int fuerza;
    private boolean herido=false,muerto=false;

    public static final int FUERZA_PORDEFECTO=5;

    public Personaje(String nombre) {
        this.nombre=nombre;
    }

    abstract public boolean retirarse() throws Exception;

    /**
     * Comrprueba si un valor es correcto para el atributo fuerza >=1 y <=10
     * @param f fuerza a establecer
     * @return
     */
    public static boolean comprobarFuerza(int f) {
        return f<=10 && f>=1;
    }
    /**
     * Establece un valor aleatorio para el atributo fuerza
     */
    protected void setFuerzaAleatoria() {
        this.fuerza=ThreadLocalRandom.current().nextInt(1, 11);
    }

    // Getters y Setters
    public int getFuerza() {
        return fuerza;
    }
    public void setFuerza(int fuerza) {
        this.fuerza = fuerza;
```

```
41      }
42      public boolean isHerido() {
43          return herido;
44      }
45      public void setHerido(boolean herido) {
46          if(muerto)
47              this.herido = false;
48          else
49              this.herido = herido;
50      }
51      public boolean isMuerto() {
52          return muerto;
53      }
54      public void setMuerto(boolean muerto) {
55          this.muerto = muerto;
56          if(muerto) setHerido(false);
57      }
58      public String getNombre() {
59          return nombre;
60      }
61
62      /**
63       *
64       * @return - String formateado de forma lineal que contiene los atributos del objeto
65       */
66      @Override
67      public String toString() {
68          return String.format("%-10s - %-6s - %-6s - %-6s",nombre,fuerza,herido,muerto);
69      }
70
71      /**
72       * Comprueba si son iguales este (this) objeto a otra objeto segun atributo nombre
73       * @param o - Objeto a comparar
74       * @return - Si se trata de un objeto clase Personaje con el atributo nombre identico
75       * devuelve 'true', en caso contrario 'false'
76       */
77      @Override
78      public boolean equals(Object o) {
79          if (this == o) return true;
80          if (o == null || getClass() != o.getClass()) return false;
```

```
81          Personaje personaje = (Personaje) o;
82          return Objects.equals(nombre, personaje.nombre);
83      }
84
85      /**
86       * Genera hash a partir del atributo nombre
87       * @return - hash (int)
88       */
89      @Override
90      public int hashCode() {
91          return Objects.hash(nombre);
92      }
93
94      /**
95       * Compara dos personajes segun su atributo 'nombre'
96       * Implementa la ordenacion natural de esta clase
97       * @param o the object to be compared.
98       * @return - un entero
99       * si int<0 entonces this.personaje va "antes" o es "menor"
100      * si int>0 entonces this.personaje va "despues" o es "mayor"
101      * si int=0 ambos son considerados iguales
102      */
103     @Override
104     public int compareTo(Personaje o) {
105         return this.nombre.compareTo(o.nombre);
106     }
107 }
108
109
110
111
```

```java
package AlejandroCastellanosExamenT11;

import java.util.*;

public class Grupo {


    //Conjunto donde guardamos Objetos de clase Personaje
    private final Set<Personaje> miembros = new HashSet<>();


    /**
     * Añade un personaje al HashSet 'miembros' del objeto de clase Grupo
     * @param p - Personaje que se añade
     * @return - 'true' si se añade correctamente,
     * 'false' si el personaje ya se encontraba dentro del conjunto
     */
    public boolean incluirPersonaje(Personaje p){
        return miembros.add(p);
    }

    /**
     * Obtiene un personaje que se encuentre en el hashset 'miembros' a partir de un nombre
     *
     * @param nombre - String con el nombre a buscar
     * @return - Personaje con mismo String en su atributo 'nombre'
     */
    public Personaje getMiembro(String nombre){
        for (Personaje miembro : miembros) {
            if (miembro.getNombre().equals(nombre)) return miembro;
        }
        return null;
    }

    /**
     * Obtiene una lista de los magos dentro de conjunto 'miembros'
     * @return - Lista de objetos clase mago
     */
    public List<Mago> dameMagos(){
        List<Mago> listaMagos = new ArrayList<>();
```

```
41          for (Personaje miembro : miembros) {
42              if(miembro instanceof Mago) listaMagos.add((Mago)miembro);
43          }
44          return listaMagos;
45      }
46
47      /**
48       * Metodo que 'retira' a todos los personajes que se encuentran
49       * en el conjunto 'miembros' llamando al metodo
50       * 'retirarse()' de cada objeto clase Personaje
51       * ---
52       * Muestra un mensaje segun se haya podido o no retirar cada personaje
53       *
54       */
55      public void retirada(){
56          for (Personaje miembro : miembros) {
57              try {
58                  miembro.retirarse();
59                  System.out.println(miembro.getNombre()+" se ha retirado");
60              } catch (Exception e) {
61                  System.out.println(miembro.getNombre()+" no se ha retirado: "+e.getMessage());
62              }
63
64          }
65      }
66
67      /**
68       * Muestra por pantalla los objetos Personaje dentro del conjunto, y sus atributos
69       * nombre, fuerza, estado herido (bool) y estado muerto (bool) correspondientes
70       * Tambien muestra la suma de sus fuerzas
71       */
72      public void mostrar(){
73          System.out.printf("%-10s - %-6s - %-6s - %-6s %n","Nombre","Fuerza","Herido?","Muerto?");
74          int fuerzaTotal = 0;
75          for (Personaje miembro : miembros) {
76              System.out.println(miembro);
77              fuerzaTotal += miembro.getFuerza();
78          }
79          System.out.println("Fuerza total del grupo: "+fuerzaTotal+"\n");
80      }
```

```java

    /**
     * Elimina los Personajes clase Troll que su atributo 'muerto' sea 'true'
     * @return - Devuelve la cantidad de Trolls eliminados
     */
    public int limpiarGrupo(){
        Iterator<Personaje> iter = miembros.iterator();
        int c = 0;
        while (iter.hasNext()){
            Personaje personaje = iter.next();
            if(personaje instanceof Troll && personaje.isMuerto()) {
                iter.remove();
                c++;
            }
        }
        return c;
    }

    /**
     * Muestra por pantalla los objetos Personaje dentro del conjunto, y sus atributos
     * nombre, fuerza, estado herido (bool) y estado muerto (bool) correspondientes
     * Tambien muestra la suma de sus fuerzas
     * ----
     * Se muestran ordenados por nombre
     * [segun su ordenacion natural (Comparable)]
     */
    public void mostrarGrupo(){
        TreeSet<Personaje> miembrosOrdenados = new TreeSet<>(miembros);

        System.out.printf("%-10s - %-6s - %-6s - %-6s %n","Nombre","Fuerza","Herido?","Muerto?");
        int fuerzaTotal = 0;
        for (Personaje miembro : miembrosOrdenados) {
            System.out.println(miembro);
            fuerzaTotal += miembro.getFuerza();
        }
        System.out.println("Fuerza total del grupo: "+fuerzaTotal+"\n");
    }

    /**
     * Muestra por pantalla los objetos Personaje dentro del conjunto, y sus atributos
```

```
121      * nombre, fuerza, estado herido (bool) y estado muerto (bool) correspondientes
122      * Tambien muestra la suma de sus fuerzas
123      * ----
124      * Se muestran ordenados por fuerza
125      * [hace uso de un lambda equivalente a un Comparator en clase anonima]
126      */
127     public void mostrarGrupoXFuerza(){
128         ArrayList<Personaje> miembrosOrdenadoFuerza = new ArrayList<>(miembros);
129         miembrosOrdenadoFuerza.sort((o1, o2) -> o2.fuerza - o1.fuerza);
130 
131         System.out.printf("%-10s - %-6s - %-6s - %-6s %n","Nombre","Fuerza","Herido?","Muerto?");
132         int fuerzaTotal = 0;
133         for (Personaje miembro : miembrosOrdenadoFuerza) {
134             System.out.println(miembro);
135             fuerzaTotal += miembro.getFuerza();
136         }
137         System.out.println("Fuerza total del grupo: "+fuerzaTotal+"\n");
138     }
139 
140 }
141 
```

```java
package AlejandroCastellanosExamenT11;

public class Mago extends Personaje{

    /**
     * Constructor que crea un mago a partir de un nombre y una fuerza
     *
     * @param nombre - Nombre del mago
     * @param fuerza - Fuerza del mago
     */
    public Mago(String nombre, int fuerza) {
        super(nombre);

        if(comprobarFuerza(fuerza))
            this.fuerza=fuerza;
        else
            this.fuerza=FUERZA_PORDEFECTO;
    }

    /**
     * Metodo que "retira" magos segun su estado de sus atributos
     *
     * @return - Devuelve 'true' si el mago esta herido (y por lo tanto, no muerto)
     * @throws Exception - Lanza exceptiones con mensaje diferente
     * en caso de que el mago este muerto o no herido
     */
    @Override
    public boolean retirarse() throws Exception {
        if(isHerido())
            return true;
        else if (isMuerto()) {
            throw new Exception("El mago esta muerto");
        }
        throw new Exception("El mago no esta herido");
    }
}

```

```java
package AlejandroCastellanosExamenT11;

public class Troll extends Personaje{

    //Atributo que cuenta el numero de trolls creados
    private static int nTrollsCreados = 1;

    /**
     * Constructor que crea un troll con un nombre predefinido segun la cantidad de trolls
     * creados y con una fuerza aleatoria
     * (no se le pasan parametros)
     */
    public Troll() {
        super("Troll_"+nTrollsCreados);
        setFuerzaAleatoria();
        nTrollsCreados++;
    }

    /**
     * Metodo que "retira" trolls segun su estado de sus atributos
     * En este caso, nunca los retira
     *
     * @return - No devuelve nada
     * @throws Exception - Siempre lanza una exception con el mensaje correspondiente
     */
    @Override
    public boolean retirarse() throws Exception {
        throw new Exception("Un troll no puede retirarse");
    }


}

```

```java
package AlejandroCastellanosExamenT11;

import java.util.HashMap;
import java.util.Map;

public class ContadorLetras {

    public static void main(String[] args) {
        String frase="Decir que Java es estupendo porque funciona con todos los sistemas operativos es como decir que el sexo anal es estupendo porque funciona con todos los géneros (Alanna)";

        //Creamos el mapa (en este caso un HashMap)
        //Clave - letra (asi que usamos tipo Character)
        //Valor - repeticiones (int)
        Map<Character,Integer> distribucion = new HashMap<>();


        //recorremos los caracteres del String a traves del array devuelto por
        // el metodo toCharArray()

        // si la clave ya se contiene, sumamos uno al valor (+1 repeticion) de la clave
        // si no, añadimos una entrada con dicha clave y valor=1
        for (char c : frase.toCharArray()) {
            if(distribucion.containsKey(c))
                distribucion.put(c,distribucion.get(c)+1);
            else
                distribucion.put(c,1);
        }

//        System.out.println(distribucion);

        //mostramos las entradas (a traves de recorrer el entrySet del mapa)
        System.out.println("DISTRIBUCION");
        for (Map.Entry<Character, Integer> entry : distribucion.entrySet()) {
            System.out.println(entry.getKey()+":"+entry.getValue());
        }



    }
```

```
40
41 }
42
```

```java
package AlejandroCastellanosExamenT11;

public class PruebasPersonaje {
    public static void main(String[] args) {

        Grupo voxMachina = new Grupo();

        Personaje m1 = new Mago("Mago-A",5);
        Personaje m2 = new Mago("Mago-B",7);
        Personaje m3 = new Mago("Zago-A",2);
        Personaje mx = new Mago("Zago-B",99);

        Personaje t1 = new Troll();
        Personaje t2 = new Troll();
        Personaje t3 = new Troll();
        Personaje t6 = new Troll();
        Personaje t5 = new Troll();
        Personaje t4 = new Troll();


        voxMachina.mostrar();

        System.out.println( voxMachina.incluirPersonaje(m1));
        System.out.println( voxMachina.incluirPersonaje(mx));
        voxMachina.incluirPersonaje(t1);
        voxMachina.incluirPersonaje(t2);

        voxMachina.mostrar();

        m1.setHerido(true);

        t1.setHerido(true);
        t1.setMuerto(true);

        t2.setMuerto(true);
        t2.setHerido(true);

        voxMachina.mostrar();

        voxMachina.incluirPersonaje(m2);
```

```
41          voxMachina.incluirPersonaje(m3);
42          voxMachina.incluirPersonaje(t3);
43          voxMachina.incluirPersonaje(t6);
44          voxMachina.incluirPersonaje(t5);
45          voxMachina.incluirPersonaje(t4);
46
47          t4.setHerido(true);
48          t5.setHerido(true);
49          m2.setMuerto(true);
50
51          voxMachina.mostrar();
52
53          voxMachina.retirada();
54          voxMachina.mostrar();
55
56          System.out.println(voxMachina.limpiarGrupo());
57          voxMachina.mostrar();
58
59          System.out.println(voxMachina.getMiembro("Mago-B"));
60          System.out.println(voxMachina.getMiembro("Troll_6"));
61          System.out.println(voxMachina.getMiembro("Experimento"));
62
63          System.out.println("\n"+voxMachina.dameMagos()+"\n\n");
64
65          voxMachina.mostrarGrupo();
66
67          voxMachina.mostrarGrupoXFuerza();
68
69
70
71
72      }
73  }
74
```